MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

2

AD-A264 017

DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

2. REPORT DATE April 12, 1993	3. REPORT TYPE AND DATES COVERED Final 17 Jun 91-16 Jun 93	
4. TITLE AND SUBTITLE Accelerating the Transfer of Technology for Implementing Domain Specific Software Architectures		5. FUNDING NUMBERS ARO 28950.1-MA
6. AUTHOR(S) Vincent P. Heuring and William M. Waite		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Colorado Boulder, Department of Electrical and Computer Engineering, Campus Box 425, Boulder, CO 80309-0425		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office, P. O. Box 12211, Research Triangle Park, NC 27709-2211		10. SPONSORING/MONITORING AGENCY REPORT NUMBER DAAL03-91-G-0203

DTIC ELECTE MAY 10 1993 S C D

11. SUPPLEMENTARY NOTES
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The purpose of the research effort was to find one or more clients within the DOD community to help evaluate the Eli system when applied to problems of interest to the DOD. We solicited information from a number of DoD agencies, and finally formed a relationship with a group at the Naval Postgraduate School (NPS). As a result of that relationship, we have developed a formal specification for the NPS language, PSDL, using the Eli system. That specification has been subsequently modified by NPS personnel whom we trained in the use of the ELi system, and the resulting PSDL translator is being used by NPS personnel.

The research also led to further modifications to the Eli system. We were able to add a "literate programming" tool to Eli that has proven to be a significant aid in organizing and explaining specifications.

14. SUBJECT TERMS			15. NUMBER OF PAGES 5
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev 2-89) Prescribed by ANSI Std Z39-18 298-102

1. Summary

Our objective was to locate and evaluate several potential customers for the Eli system,[1] demonstrate how this technology could improve their productivity, and provide the information and training needed for them to make effective use of it. We hoped to complete one pilot project in order to gain credibility for the technology so that another of the potential customers would be willing to participate in *and support* a second pilot project.

We solicited information from a number of DoD agencies, and finally formed a relationship with a group at the Naval Postgraduate School (NPS). This group is developing an approach to rapid prototyping of real-time systems, using a language called PSDL.[2] They needed a translator from PSDL to Ada that could be constructed quickly, and would be flexible enough to accommodate changes in both PSDL and the mapping from PSDL to Ada. Eli is ideally suited to this task, and the NPS group was not satisfied with the other tools they had investigated.

The cooperation began in the fall of 1991, when the NPS group provided us with their current grammar for PSDL and an account on their computer. That grammar and reference 2 were the only available description of PSDL. Our first task was to understand PSDL and try to formalize that understanding using Eli. This task involved studying the documents, formulating hypotheses, and verifying those hypotheses via electronic mail. Our NPS contact was Lt. Charles Altizer, a naval instructor employed on the project.

By late spring of 1992, we had completed the first task. The formalization of PSDL had uncovered several inconsistencies in the language, and these had been corrected in consultation with the NPS group. We had implemented the Eli system on the NPS machine, and Lt. Altizer had begun to familiarize himself with the documentation. The NPS group invited Prof. Waite to spend a week in Monterey to provide hands-on instruction in the use of Eli. Due to scheduling conflicts, this visit could not be made until the week of July 13-17.

During his visit, Prof. Waite demonstrated Eli. Then he and Lt. Altizer went through a case study involving a part of the translation, with Prof. Waite demonstrating the implementation techniques in the course of creating that part of the translator. After that part had been completed, Lt. Altizer embarked upon another part under Prof. Waite's supervision. Finally, Lt. Altizer implemented yet a third part of the translator with only minimal aid.

Lt. Altizer continued the translator implementation during the fall of 1992, with consultation on problems via electronic mail. In January of 1993, he spent a week in Boulder working with several members of the Eli group on a particularly subtle part of the translation.

We believe that the pilot study involving NPS was successful. The technology was shown to be appropriate for the problem, and the NPS group was able to effectively use that technology to solve the problem. Feedback on certain difficulties with large specifications led to incorporation of a new "literate programming" tool into the Eli system. This tool is proving to be a significant aid in organizing and explaining specifications.

2. The PSDL Project

PSDL is a specification language that supports rapid prototyping of real-time systems. A user specifies the desired real-time system via a set of *operators*, where each operator is either implemented in Ada or described in PSDL. Properties and assertions can be associated with each operator and used to search a software data base for appropriate implementations. A graphical editor is used to construct and update PSDL descriptions presented in the form of directed graphs.

The user's specification is reduced to a flat, directed graph by other components of the rapid-prototyping system. Each node in the graph represents an instance of a primitive operator that is available in Ada and stored in the software data base. Each arc in the graph represents a stream of data items. This must be converted to an Ada specification that describes the interconnections, triggering conditions, input/output operations and module invocations necessary to simulate the specified real-time system. The resulting Ada specification is then compiled to obtain a running simulator.

Conversion of the flat graph to an Ada specification is a compilation problem: The graph must be checked for consistency, associations between formal parameters and arguments must be established, and Ada text produced. This problem is not well defined because the entire system is evolving. Despite its ill-defined nature, it must be solved if progress is to be made. Without the ability to convert flat graphs to Ada specifications, it is impossible to create running simulators.

The notation for describing the flat graph is quite stable. Its structure and context conditions will probably not change radically. The appropriate Ada specification for a given flat graph is less certain. Thus the translator that converts a graph to an Ada specification must be easily altered when changes to the form of the output specification are proposed.

During the first part of this project, our group constructed a specification to describe the lexical and syntactic structure of PSDL. We also provided specifications for context conditions on operators, but did not deal with the data type identifiers. All of this work was done in consultation with the NPS group, since some of the language definition existed only in the minds of the designers. Thus, in addition to providing the basis of a translator, these specification files became the official description of PSDL itself.

There is no written description of the mapping from PSDL to Ada, nor is there any but a fragmentary description of the PSDL typing mechanism. We could not, therefore, provide much support for these aspects of the language. We therefore used the specifications that we had developed as a teaching tool and a point of departure for transferring the technology to the NPS group. Our strategy was to use the constructs we had described as case studies, having the NPS group explain the equivalent Ada specification for each and then implementing the translation rules to produce that equivalent Ada specification. Prof. Waite implemented the first such case study, carefully explaining the rationale to Lt. Altizer as he wrote the rules, and answering all of the questions that came up in the process. The second case study was implemented by Lt. Altizer, with Prof. Waite asking questions and providing guidance. Lt. Altizer then implemented the third case study with minimal interaction from Prof. Waite.

The remainder of the project was carried out by Lt. Altizer with consultation via electronic mail. At one point the sophistication of the translation was pushing the capabilities of Eli and Lt. Altizer spent a week in Boulder working with several members of the group. This visit resulted in new perspectives on one component of Eli, and an elegant solution to the problem of mapping PSDL types to Ada.

3. Changes to Eli

Several changes to the Eli system and its documentation were made as a result of our experiences with PSDL and the Naval Postgraduate School. The most important was our inclusion of a tool that allows us to combine specifications of different types in a single file. A specification that solves a problem like the PSDL translation problem can be understood only if it is decomposed according to task: Each subtask is specified separately. Unfortunately, a single task usually involves several different kinds of specification, each of which is relatively small. Thus the number of small files increases rapidly under this sort of decomposition. When a change is needed, related changes must be made in a number of files, and maintenance of the specification becomes a nightmare.

The tool we used to solve this problem is called *FunnelWeb*. It was developed in Australia, and is an implementation of Knuth's ''literate programming'' paradigm.[3,4] Unlike most such tools, FunnelWeb allows one to produce many different files from a single description. That means we can gather together all of the small specifications of different kinds that relate to a single task into a single file for that task. When a change is required, only one file is involved instead of many.

FunnelWeb also gives Eli the ability to construct formatted text files from the specifications. Using FunnelWeb, the PSDL specification can be written in such a way as to serve both as a human-readable definition of the language and mapping, and as the source from which the processor itself is generated. This makes it considerably easier to keep the language definition and the translator consistent, and to guarantee a complete language definition.

The other changes occurred in the documentation. A number of clarifications were necessary, and a major re-orientation of the discussion of overload resolution was required. PSDL is the first language that we had encountered that needed to distinguish many instantiations of a particular abstract data type symbolically. (This was necessary in order to produce the Ada specification.) Although we were able to devise an elegant solution with the existing mechanisms, that solution was virtually impossible to see given the available documentation.

4. References

1. R. W. Gray, V. P. Heuring, S. P. Levi, A. M. Sloane and W. M. Waite, 'Eli: A Complete, Flexible Compiler Construction System', *Communications of the ACM*, **35**, 121-131 (February 1992).

2. Luqi, V. Berzins and R. T. Yeh, 'A Prototyping Language for Real-Time Software', *IEEE Transactions on Software Engineering*, **14**, 1409-1423 (October 1988).

3. D. E. Knuth, 'Literate Programming', *The Computer Journal*, **27**, 97-111 (1984).

4. L. M. C. Smith and M. H. Samadzadeh, 'An Annotated Bibliography of Literate Programming', *SIGPLAN Notices*, **26**, 14-20 (January 1991).